# Cours 2 Microprocesseurs

Jalil Boukhobza
LC 206
boukhobza@univ-brest.fr

1

## Chemin de données

Font l'objet de ce cours:

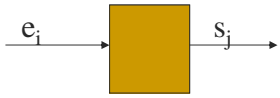
- Les portes logiques et circuits combinatoires
- Le traitement de quelques opérations arithmétiques (UAL)
- Circuits séquentiels, registres et mémorisation
- l'interconnexion des différents éléments du chemin de données

*Définition*: ensemble des composants requis pour l'exécution des diverses instructions (banc de registres, PC, UAL, etc.)

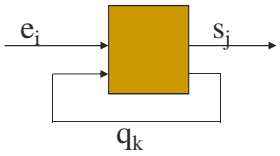
2

# Circuits logiques

- **Circuit logique combinatoire** : l'état des sorties $s_j$ dépend uniquement de l'état courant des entrées $e_i$.
  $s_j=f(\ldots, e_i, \ldots)$




- **Circuit logique séquentiel** : l'état des sorties $s_j$ dépend de l'état courant des entrées $e_i$ et de l'état $(\ldots, q_k, \ldots)$ de la machine
  $s_j=f(\ldots, e_i, \ldots, q_k, \ldots)$
  $q_k=g(\ldots, e_i, \ldots, q_k, \ldots)$




3

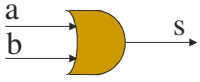
L'algèbre de Boole

# Opérateur OU

- Identité : 0+a=a
- Nullité : 1+a=1
- Idempotence : a+a=a
- Commutativité : a+b=b+a
- Associativité : (a+b)+c=a+(b+c)

Porte OU (OR)

s=a+b




| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

4

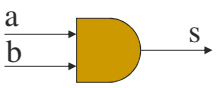
L'algèbre de Boole

## Opérateur ET

- Identité : 1.a=a
- Nullité : 0.a=0
- Idempotence : a.a=a
- Commutativité : a.b=b.a
- Associativité : (a.b).c=a.(b.c)

Porte ET (AND)

s=a.b




| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

5

L'algèbre de Boole

## Lois de composition

- Distributivité

  a.(b+c)=a.b+a.c  a+(b.c)=(a+b).(a+c)
- Absorption

  a.(a+b)=a  a+a.b=a
- De Morgan

  $\overline{a.b}=\overline{a}+\overline{b}$  $\overline{a+b}=\overline{a}.\overline{b}$

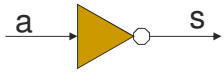
6

Logique combinatoire

## Portes logiques unaires

Porte NON (NOT) ou inverseur

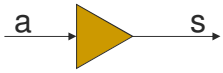
$s = \overline{a}$




| a | s |
|---|---|
| 0 | 1 |
| 1 | 0 |

Porte OUI ou répéteur

$s = a$




| a | s |
|---|---|
| 0 | 0 |
| 1 | 1 |

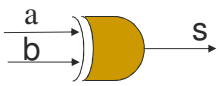
7

Logique combinatoire

## Portes logiques binaires (suite)

Porte OU exclusif (XOR)

$s=a \oplus b$




| a | b | s |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$s=a \oplus b=\overline{a}.b+a.\overline{b}$

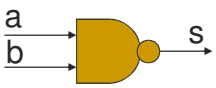
8

Logique combinatoire

## Portes logiques binaires (suite)

Porte NON-ET (NAND)

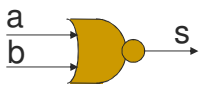
$s=\overline{a.b}$




| a | b | s |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Porte NON-OU (NOR)

$s=\overline{a+b}$




| a | b | s |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

9

## L'addition en nombres non signés

$$\begin{array}{cccc} & 1 & 1 & 1 \\ & 1 & 0 & 1 \\ \hline 1 & 1 & 0 & 0 \end{array}$$

Retenue sortante

Étude de l'architecture de l'additionneur à retenue propagée.

10

# L'additionneur à retenue propagée

- Addition sur 1 bit, somme variant de la valeur 0 à 2:

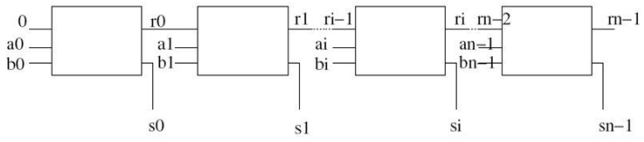
$$somme(poids\ faible) = a \oplus b = s(0)$$
$$somme(poids\ fort) = a.b = retenue = r(0)$$

- Addition sur n bits:

$$s(i) = a(i) \oplus b(i) \oplus r(i-1)$$
$$r(i) = a(i).b(i) + r(i-1).(a(i) \oplus b(i))$$
$$avec\ i \in [0..n-1[$$

11

# Additionneur complet (4 bits) à retenue propagée




Additionneur à retenue propagée

Problème de **temps de propagation**: circuit lent car il dépend du temps de propagation de la retenue.

12

# Addition avec des nombres signés

- En complément à 2 : pas de traitement particulier pour le bit de signe.

| | | | | | |
|---|---|---|---|---|---|
| | 1 | 1 | 0 | 0 | (–4) |
| | 1 | 1 | 0 | 1 | (–3) |
| 1 | 1 | 0 | 0 | 1 | (–7) |

La retenue du dernier étage *n'est pas synonyme de dépassement de capacité.*

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | (4) |
| | 0 | 1 | 0 | 1 | (5) |
| 0 | 1 | 0 | 0 | 1 | (–7) |

Pas de dépassement de capacité si $x + y \in \left[-2^{n-1}-1,\quad 2^{n-1}\right]$

13

# Cas de débordement

- Addition de nombres positifs (res ≥ $2^{n-1}$)

| | | | | | |
|---|---|---|---|---|---|
| | 0 | 1 | 1 | 1 | (7) |
| | 0 | 0 | 0 | 1 | (1) |
| 0 | 1 | 0 | 0 | 0 | (–8) |

- Addition de nombres négatifs (res ≤ $-2^{n-1}-1$)

| | | | | | |
|---|---|---|---|---|---|
| | 1 | 1 | 0 | 0 | (–4) |
| | 1 | 0 | 1 | 0 | (–6) |
| 1 | 0 | 1 | 1 | 0 | (6) |

- Mise en équation du débordement (overflow):

$$overflow = a.b.\bar{s} + \bar{a}.\bar{b}.s$$

14

## Multiplication des nombres non signés

- A et B opérandes sur N bits

$$A = A_{n-1}*2^{n-1} + A_{n-2}*2^{n-2} + A_{n-3}*2^{n-3} + \ldots + A_1*2^1 + A_0*2^0$$
$$B = B_{n-1}*2^{n-1} + B_{n-2}*2^{n-2} + B_{n-3}*2^{n-3} + \ldots + B_1*2^1 + B_0*2^0$$

- Le produit A.B nécessite *2n* bits

$$A.B = A*B_{n-1}*2^{n-1} + A*B_{n-2}*2^{n} + \ldots + A*B_1*2^1 + A*B_0*2^0$$

- *A.B* est une somme de produits partiels $P_i$:

$$P_i = A.B_i.2^i$$

$P_i$ est obtenu par décalage si $B_i$=1.

15

## Exemple

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 0 | 1 | (13) |
| | | | | 1 | 0 | 0 | 1 | (9) |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | (*P*0) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | • | (*P*1) |
| 0 | 0 | 0 | 0 | 0 | 0 | • | • | (*P*2) |
| 0 | 1 | 1 | 0 | 1 | • | • | • | (*P*3) |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | (*P*3) |

Pour des multiplications ou divisions de nombre non signés par $2^n$, on procède par décalages;
- vers la gauche pour la multiplication
- vers la droite pour la division

16

## Multiplication de nombres signés

- En complément à 2, B s'écrit:

  $B=-B_{n-1}*2^{n-1}+B_{n-2}*2^{n-2}+\ldots+B_1*2^1+B_0*2^0$

- Le produit A.B s'écrit:

  $A.B=-A*B_{n-1}*2^{n-1}+A*B_{n-2}*2^{n-2}+\ldots+A*B_1*2^1+A*B_0*2^0$

**même principe mais une soustraction à la place d'une addition.**

17

## Exemple

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 0 | 1 | (−3) |
| | | | | 0 | 1 | 0 | 1 | (5) |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | (*P*0) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | • | (*P*1) |
| 1 | 1 | 1 | 1 | 0 | 1 | • | • | (*P*2) |
| 0 | 0 | 0 | 0 | 0 | • | • | • | (*P*3) |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | (−15) |

18

# Autre exemple

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 0 | 1 | $(-3)$ |
| | | | | 1 | 1 | 0 | 1 | $(-3)$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | $(P0)$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | • | $(P1)$ |
| 1 | 1 | 1 | 1 | 0 | 1 | • | • | $(P2)$ |
| 0 | 0 | 0 | 1 | 1 | • | • | • | $(-P3)$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | $(9)$ |

19

# Une courte introduction au flottant

Notation exponentielle pour la représentation des flottants

Mantisse * Base $^{\text{Exposant}}$

- **Mantisse** : précision du nombre (m). La virgule se trouve par défaut après le premier bit le plus à gauche du nombre, en général pas représenté car est égal à la valeur 1 (cas des nombres normalisé).
- **Exposant** : ordre de grandeur et place de la virgule dans la mantisse (e). L'exposant est donné en représentation biaisée afin qu'il soit toujours positif. Un exposant sur 8 bits varie avec la représentation biaisée de 0..255. Pour obtenir la vraie valeur du biais, on doit soustraire 127, l'espace de variation est alors de –127 à +128.

On pose alors e=[E]-127 et $m = (1 + \frac{[M]}{2^{23}})$ avec E et M correspondant à la représentation du flottant (+bit du signe).

20

## Exemple

-54,625

$54=(110110)_2$

$0,625=(0,101)_2$ …. 0,625*2=1,25

0,25*2 =0,5

0,5*2 =1,0 → STOP

$54,625=(110110,101)_2$ = 1 10110101*$2^5$

Biais=$2^{(\text{nombre de bits pour l'exposant biaisé } -1)}$ -1=$2^{(8-1)}$-1=127

Exposant biaisé=exposant réel + biais=5+127=132
=$(10000100)_2$

Résultat:

-54,625=(1 10000100 10110101000000000000000)$_2$

21

## Format IEEE 754

Simple précision (double précision)

- s: 1 bit de signe
- E: 8 bits pour l'exposant (11 bits)
- M: 23 bits pour la mantisse (52 bits)

un flottant normalisé s' évalue de la manière suivante:

$$(-1)^s * (1 + \frac{[M]}{2^{23}}) * 2^{[E]-127}$$

Par exemple:

1.1010001 * $2^{(10100)}$ =(1+1/2+1/$2^3$+1/$2^7$)*$2^{20}$= 1. 6328125 * $2^{20}$

Biais= 20+127=147= $(10010011)_2$

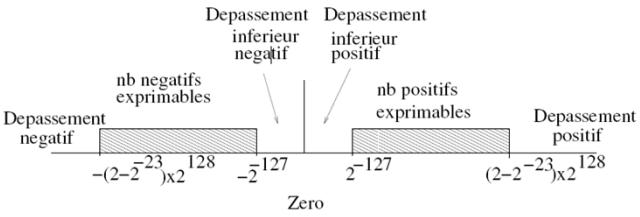
Est représenté par:

0 10010011 10100010000000000000000

22

# Espace de nombres représentables

- Sur 32 bits, nous avons $2^{32}$ valeurs représentables.
- En notation complément à 2 tous les entiers de $-2^{31}$ à $2^{31} - 1$.
- En représentation flottante, les nombres ne sont pas régulièrement espacés




23

# Codages spéciaux

- **Zéros** : pas de représentation normalisée
  e=-127, m=0 => E= 00000000, M = 0…0
  2 représentations du zéro (signe)
- **Infinis**: -∞ et +∞
  e=128, m=0, => E=11111111, M=0…0
- ***NaN*** (Not a Number) utilisé pour représenter les erreurs
  e=128 et et m≠0.

24

## Les opérations en flottant

Déroulement

1. Aligner les exposants sur l'exposant du plus grand nombre
2. Opération
3. Normalisation du résultat → pb d'arrondi

25

## Indicateurs fournis par les opérateurs

Indicateurs supplémentaires fournis par les opérateurs:

- I : résultat inexact, mantisse arrondie
- V : opération invalide,
- Z : division par zéro
- O : Overflow dépassement de capacité, supérieur à $1,1\ldots1 * 2^{127}$ en valeur absolue
- U : Underflow dépassement de capacité, inférieur à $2^{-126}$ en valeur absolue

26

## Unité Arithmétique et Logique (UAL)

Une UAL peut être définie pour réaliser:

- des opérations arithmétiques sur des entiers positifs, négatifs ou flottants
- des opérations de comparaison
- de décalages
- des opérations logiques

Les drapeaux générés par l'UAL le sont généralement pour le dépassement de capacité, pour le cas de résultat nul, pour le cas de résultat négatif ou pour le cas de résultats erronés dus à des erreurs d'arrondis ou `a des opérations non valides.

27

## Éléments de mémorisation

- Les **registres** sont des éléments *séquentiels* qui sont activés par une *horloge* et permettent de mémoriser des valeurs (état interne).
- Les registres sont constitués d'éléments de mémorisation élémentaires généralement des bascules D ou des latchs D.
- La bascule D est synchronisée sur front d'horloge tandis que le latch D est synchronisée sur niveau d'horloge (circuit plutôt asynchrone dans lequel on ne maîtrise pas complètement les instants de changement de la sortie).

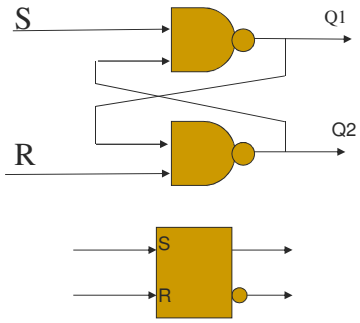
28

# Latch RS

- Un latch RS (Reset-Set) est un circuit séquentiel asynchrone.

$Q1=\overline{S.Q2}$

$Q2=\overline{R.Q1}$

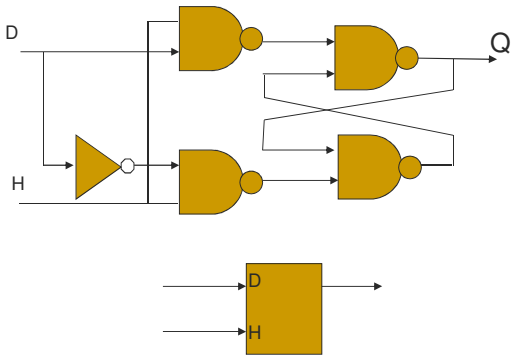



| R | S | Q1- | Q2- | Q1 | Q2 |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | X | X | 0 | 1 |
| 1 | 0 | X | X | 1 | 0 |
| 0 | 0 | X | X | 1 | 1 |

29

# Latch D (verrou)

- Un latch D (verrou ou latch) est un circuit séquentiel synchrone.
- La sortie Q recopie l'entrée D lorsque le signal d'horloge H est actif.

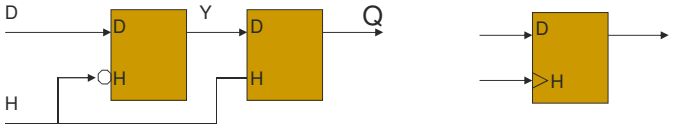


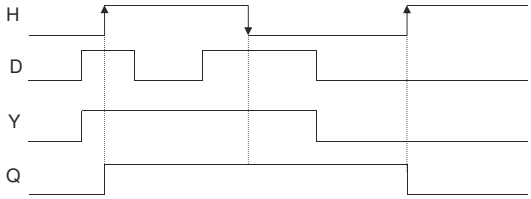

| D | H | Q |
|---|---|---|
| 0 | 0 | Q- |
| 1 | 0 | Q- |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

30

# Bascule D (flip-flop)

- Une bascule D (flip-flop) est un circuit séquentiel synchrone.
- La sortie Q recopie l'entrée D lorsque le signal d'horloge H passe de l'état 0 à l'état 1 (front montant).
- Structure maître-esclave




31

# Table de vérité des bascules/latch D

| Entrée $D$ | Horloge | etat interne | sortie $D$ flip-flop | sortie $D$ latch |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 (mémorisation) | 0 (mise à 0) |
| 0 | 1 | 1 | 1 (mémorisation) | 0 (mise à 0) |
| 1 | 1 | 0 | 0 (mémorisation) | 1 (mise à 1) |
| 1 | 1 | 1 | 1 (mémorisation) | 1 (mise à 1) |
| 0 | ↑ | 0 | 0 (mise à 0) | 0 (mémorisation) |
| 0 | ↑ | 1 | 0 (mise à 0) | 1 (mémorisation) |
| 1 | ↑ | 0 | 1 (mise à 1) | 0 (mémorisation) |
| 1 | ↑ | 1 | 1 (mise à 1) | 1 (mémorisation) |
| 0 | ↓,0 | 0 | 0(mémorisation) | 0 (mémorisation) |
| 0 | ↓,0 | 1 | 1(mémorisation) | 1 (mémorisation) |
| 1 | ↓,0 | 0 | 0(mémorisation) | 0 (mémorisation) |
| 1 | ↓,0 | 1 | 1(mémorisation) | 1 (mémorisation) |

32

# Quelques variantes de bascules D

- Bascule D avec autorisation de chargement synchrone

| CLK | EN | état futur $Q^+$ | fonction réalisée |
|---|---|---|---|
| 0,1,↓ | X | Q | Maintien |
| ↑ | 0 | Q | Maintien |
| ↑ | 1 | D | Recopie de $D$ |

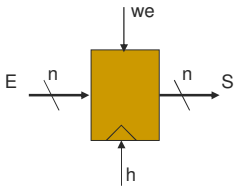
- Bascule D avec reset asynchrone

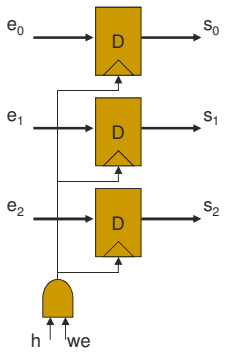
| CLK | reset | état futur $Q^+$ | fonction réalisée |
|---|---|---|---|
| 0,1,↓ | 1 | 0 | Mise à 0 |
| ↑ | 1 | 0 | Mise à 0 prioritaire |
| ↑ | 0 | D | Recopie de $D$ |

33

# Registres

- N bascules D sont associées pour enregistrer un mot (…de N bits).
- Une entrée « *write enable* », optionnelle, permet de valider ou non le signal d'horloge




34

# Variantes au niveau du registre

- Synchronisation de données en mode parallèle
  Après une impulsion d'horloge $Q_i^+ = D_i$
- Registre à décalage (avec bascules actives sur front uniquement)
  - Décalage à droite $Q_i = D_{i+1}$

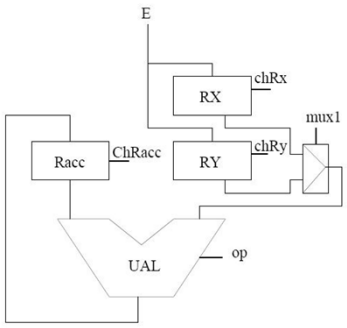
| Entrée G | $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ |
|---|---|---|---|---|---|
| t | 1 | 0 | 1 | 1 | 0 |
| t+1 | E G | 1 | 0 | 1 | 1 |

  - Décalage à gauche $Q_{i+1} = D_i$

| $Q_0$ | $Q_1$ | $Q_2$ | $Q_3$ | $Q_4$ | Entrée D |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | t |
| 0 | 1 | 1 | 0 | E D | t+1 |

35

# Éléments pour l'interconnexion des composants




- Les aiguillages peuvent être réalisés en utilisant des multiplexeurs ou des bus 3 états.

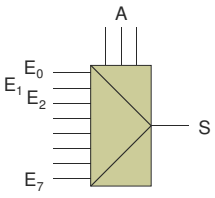
36

# Les multiplexeurs

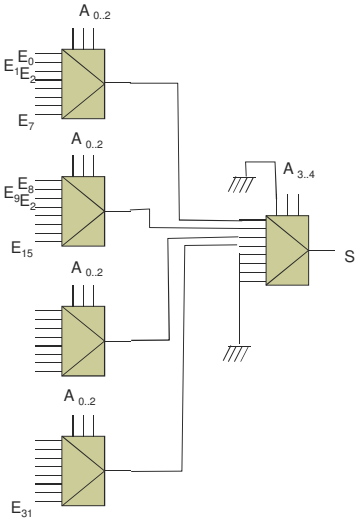
- Circuit qui accepte plusieurs signaux logiques en entrées (données) et n'autorise qu'un seul signal de sortie. L'entrée transférée en sortie est donnée par l'adressage.
- Équation générale : $S = \sum_{j=0}^{j=n} E_j.(A = j)$
  Possibilité d'avoir des entrées de validation.
- Exemple : Multiplexeur 8 vers 1 et réalisation d'un multiplexeur 32 vers 1 de manière hiérarchique.
- Sur le même principe, on peut définir un multiplexeur de mots.
- Le démultiplexeur définit le circuit réalisant l'opération inverse.

37

# Multiplexeur (2)



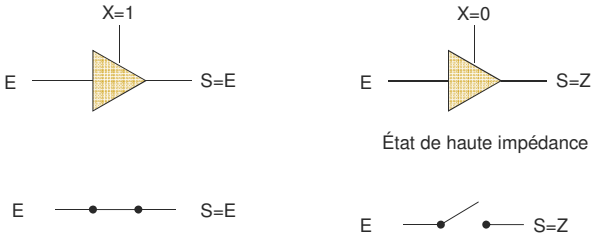

Mux 8 vers 1




38

# Portes 3 états (Tristate)

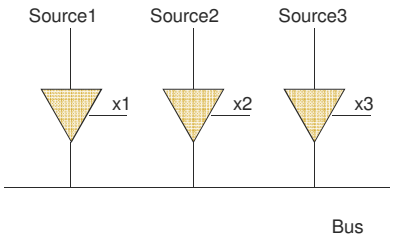
- Introduire des portes 3 états pour gérer les conflits d'écriture sur des bus.




39

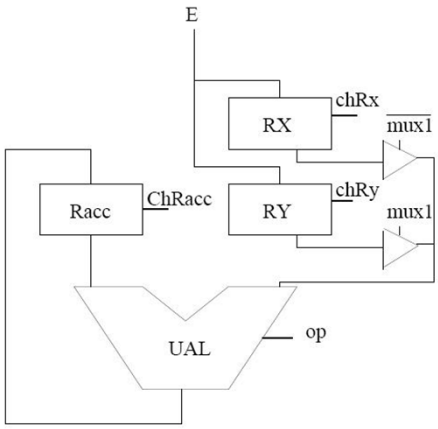
# Écriture sur un bus

- Une seule source est autorisée à un instant donné à écrire sur le bus.




40

# Utilisation des portes trois états




41